ZIW AF 2193

OIPE JC101
MAY 2 3 2005
PATENT & TRADEMARK OFFICE

Docket No.: 042390.P10802

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Arch D. Robison Application No.: 09/844,345 Filed: April 27, 2001 For: PRUNING LOCAL GRAPHS IN AN INTER-PROCEDURAL ANALYSIS SOLVER	Examiner: Insun Kang Art Group: 2193

APPEAL BRIEF

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicant submits the following Appeal Brief pursuant to 37 C.F.R. § 41.37 for consideration by the Board of Patent Appeals and Interferences. Applicant also submits herewith our check number 31741 in the amount of $500.00 to cover the cost of filing the opening brief as required by 37 C.F.R. §41.20(b). Please charge any additional fees or credit any overpayment to our deposit Account No. 02-2666. A duplicate copy of the Fee Transmittal is enclosed for this purpose.

```
05/24/2005 AWONDAF1 00000069 09844345
01 FC:1402                    500.00 OP
```

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, Intel Corporation.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences known to the appellants, the appellants' legal representative, or assignee, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-38 of the present application are pending and remain rejected. The Applicant hereby appeals the rejection of claims 1-38.

IV. STATUS OF AMENDMENTS

The Applicant filed an amendment on January 20, 2005, in response to a Final Office Action issued by the Examiner on December 22, 2004. In response to the January 20, 2005 amendment, the Examiner issued an Advisory Action on April 6, 2005. The Applicant filed a Notice of Appeal on March 22, 2005.

V. SUMMARY OF CLAIMED SUBJECT MATTER

1. Independent claims 1, 15, and 29.

One embodiment of the invention is a pruning technique to reduce the size of the local graphs used in inter-procedural analysis (IPA) solver for separately compilable software entities[1].

The local graph pruning module 142 and the IPA solver 144 are used to optimize the IPA process for separately compilable software entities. The approach is based on a lattice-theoretic framework for IPA of programs consisting of more than one separately compiled translation units. Local inter-procedural problems are constructed for each

[1] See Specification, paragraph [0023].

translation unit, reduced, and merged together into a global problem to be solved. Local solutions are derived from the global solution, and used to optimize each translation unit when it is recompiled. All problems and solutions are formulated over lattices in a way that allows the framework to automatically determine when files are either recompiled (for correctness) or optionally recompiled (for possible improvement)[2].

A local problem p_i is created for each translation unit i. For N translation units, there are N local problems. The set of all possible problems form a partial order. The set of all possible solutions form a partial order. The local problems p_I's are then represented by local graphs. Then, the local graphs are then pruned. The pruned local graphs provide reduced storage requirements and speed up input/output (I/O) transfer time when files are written and read from disk[3].

The usual form for representing a problem and a solution is a directed graph. A directed graph is a set of vertices and a set of directed edges. Each edge connects its tail vertex to its head vertex. An edge from vertex u to vertex v is denoted $u \rightarrow v$. Each vertex of the graph has a lattice value, and each edge has a monotone lattice transfer function. Value of a vertex u is denoted as u.val or val(u). Transfer function of an edge e is denoted as e.func or func(e)[4].

The local graphs are pruned or reduced and then merged together to form a global graph. The merging process in the IPA solver merges vertices with identical names. Anonymous vertices are never merged. Duplicate edges between the same pair of named vertices are merged into a single edge between the same vertices with a transfer function that is the lattice-meet of the transfer functions for the duplicates. A greatest fix-point global solution is then computed for the global graph, and some values for named vertices of the global are reported back to the local compilations. The greatest fix-point solution is a mapping of vertices to values[5].

Local graphs are pruned as follows. First, a shrinking transform shrinks the local graphs. The shrinking transform scans some special cases to perform preliminary graph

[2] See Specification, paragraph [0034].
[3] See Specification, paragraph [0044]; Figure 2.
[4] See Specification, paragraph [0042].
[5] See Specification, paragraph [0047].

reduction. A use attribute is associated to each vertex in each of the local graphs. The use attribute for a vertex v, when asserted, indicates that there is some named vertex u such that there is an edge u → v. An affect attribute is associated to each vertex in each of the local graphs. The affect attribute for a vertex u, when asserted, indicates that there is some named vertex v such that there is an edge u → v[6].

Then, a subgraph of each of the local graphs is pre-solved. The subgraph includes subgraph edges. Each of the subgraph edges connects a tail vertex to a head vertex where the tail vertex has a negated use attribute. Pre-solving the subgraph is solving a greatest fix-point problem for the subgraph of the local graph, yielding a solution SOL(w), and assigning w.val := SOL(w) for each vertex w that is the head or tail of an edge in the subgraph. Doing so changes the values of the vertices such that the contribution of the local graph problem to the global graph problem is the same even if the subgraph's edges and anonymous vertices are removed. The purpose of the pre-solver is to push the constraints from parts of the graph that will be omitted from the global graph problem, which will be solved by the IPA solver, to parts of the graph that will be included in the global problem[7].

The shrinking transform is applied again to the local graphs. This is because the new values for the vertices may enable more opportunities for shrinking. In particular, the pre-solver may have set more vertices' values to bottom, thus permitting more edges to be removed by the shrinking transform[8].

Then, final edges to be sent to the IPA solver are determined. The final edges are those edges of the form u → v where the affect attribute of u and the use attribute of v are asserted. Next, the final vertex values to be sent to the IPA solver are determined. The final vertex values are those values that are different from the lattice top because values of top contribute no information and therefore can be elided. Since most of the vertex values are top, this saves space in the stored representation of the graph. The final edges and the final vertex values form the pruned local graphs. Next, the final edges and final vertex values as pruned local graphs are sent to the IPA solver[9].

[6] See Specification, paragraph [0051]; Figure 3.
[7] See Specification, paragraph [0052].
[8] See Specification, paragraph [0053].
[9] See Specification, paragraph [0054].

VI. <u>GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL</u>

1. Claims 1-38 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,077,313 issued to Ruf ("<u>Ruf</u>").

VII. <u>ARGUMENTS</u>

A. <u>Claims 1-38 Are Not Anticipated by Ruf</u>

In the final Office Action, the Examiner rejected claims 1-38 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,077,313 issued to Ruf ("<u>Ruf</u>"). Applicant respectfully traverses the rejection and contends that the Examiner has not met the burden of establishing a prima facie case of anticipation.

To anticipate a claim, the reference must teach every element of the claim. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." <u>Vergegaal Bros. v. Union Oil Co. of California</u>, 814 F.2d 628, 631, 2 USPQ 2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the...claim." <u>Richardson v. Suzuki Motor Co.</u>, 868 F.2d 1226, 1236, 9 USPQ 2d 1913, 1920 (Fed. Cir. 1989).

Applicant contends that <u>Ruf</u> does not disclose, either expressly or inherently, (1) pruning, (2) local graphs representing local problems that correspond to separately compilable components, and (3) values of each of the local graphs form a lattice under a partial ordering.

<u>Ruf</u> merely discloses a type partitioned dataflow analyses. Type partitioned dataflow analysis performs a dataflow analysis of a program by partitioning the dataflow analysis into phases (<u>Ruf</u>, col. 6, lines 61-65). The partitioning is performed based on a dependence relation over types representing run-time. A dependence analysis determines the dependence relation among types of the corrupted type relation (<u>Ruf</u>, col. 7, lines 20-22). Dependence analysis represents the dependence relation in the form of a dependence graphs (<u>Ruf</u>, col. 9, lines 39-41). Partitioning algorithm module collapses each strongly-connected component of dependence graph into a single node (<u>Ruf</u>, col. 9, lines 66-67; col. 10, line 1). The resulting dependence graph is a directed acyclic graph (DAG) corresponding to a partial ordering of type representatives (<u>Ruf</u>, col. 10, lines 8-9).

First, Ruf merely discloses transforming a dependence relation by merging depending types in the dependence relation into a single type. Transforming and/or merging is not the same as pruning. Pruning is in essence reducing whereas merging is combining. The two processes are different. Second, Ruf does not disclose local graphs representing local problems corresponding to separately compilable components. Ruf merely discloses a dependence graph. Each type in the dependence relation is represented by a vertex or node in the graph (Ruf, col. 9, lines 42-43). A directed edge represents the dependence between each pair of types (Ruf, col. 9, lines 43-45). Since the vertex/node represents a type, it does not correspond to separately compilable components in a software program. Third, Ruf does not disclose values of the local graph form a lattice under a partial ordering. Ruf merely discloses a directed acyclic graph (DAG) which represents a collapsed dependence graph (Ruf, col. 10, lines 6-9).

In response to the above arguments, the Examiner states that the instant specification recites, "Local inter-procedural problems are constructed for each translation unit, reduced, and merged together into a global problem to be solved (0034 page 8)" (Final Office Action, page 9, item 1). Applicant respectfully disagrees. The cited paragraph merely states that the local problems are reduced and merged together into a global problem. Claim 1 recites "pruning local graphs representing local problems", which is the "reduced" phase of the cited paragraph. Claim 1 does not recite the "merged" phase of the cited paragraph.

The Examiner further states that Ruf specifically states that "Partitioning algorithm module . . . may merge suitable type representatives of the ordered dependence relation into a single type representative to help reduce execution time and/or memory space costs in performing the dataflow analysis (col. 10 lines 47-60)". The Examiner then concludes that "[t]herefore, merging includes reducing in Ruf's partitioning algorithm, accordingly, it helps 'reduce peak storage requirements for the dataflow analysis (col. 10 lines 31-47). Applicant respectfully disagrees. Reducing execution time or peak storage requirements is not the same as pruning the local graphs. In fact, Ruf clearly states that this reduction of execution time is achieved by merging the representatives of the ordered dependence relation into a single type representative (Ruf, Col. 10, lines 47-51). Furthermore, type representatives of the ordered dependence relation are not local graphs that represent local problems corresponding to separately compilable software components.

The Examiner further states that a type in Ruf corresponds to a separately compilable component because it also represents a function (Final Office Action, page 9, item 2). Applicant respectfully disagrees. First, Ruf merely states that dataflow analysis dependencies may be determined for types representing functions (Ruf, Col. 3, lines 1-4). Ruf does not disclose that the local graphs represent the local problems corresponding to separately compilable components. Dataflow analyses merely generate a model of every program quantity of interest (Ruf, Col. 1, lines 23-26). They do not prune local graphs. Second, a function is not a separately compilable component. In fact, Ruf's dataflow analysis is used to approximate the expected run-time behavior of a program (Ruf, Col. 1, lines 15-17). The functions, if present, exist in the same program, not as separately compilable components. A function, by itself, cannot be compiled separately within the context of the dataflow analysis.

The Examiner further contends that "the applicant simply states that Ruf does not disclose the limitation, values of the local graph form a lattice under a partial ordering, in the claims and fails to show why the limitation is different from the teachings of Ruf." (Final Office Action, Page 10, item 3). Applicant respectfully disagrees. It is the Examiner's burden to show that the prior art reference discloses the claimed invention. Here, the Examiner has not met this burden of showing that Ruf discloses values of the local graphs form a lattice under a partial ordering.

The Examiner further states that Ruf's directed acyclic graph (DAG) corresponds to a partial ordering of type representatives (Ruf, Col. 10, lines 1-10). However, Ruf's DAG does not correspond to values of the local graphs. Ruf's DAG represents a dependence graph. A dependence graph represents dependencies among types using the mapping of program quantities to types as defined by type relation (Ruf, Col. 10, lines 7-10; Col. 9, lines 28-32). In contrast, the local graphs correspond to separately compilable components, which do not contain dependencies among types.

In the Advisory Action dated April 6, 2005, the Examiner states that:

> "In response, Ruf specifically states that "Partitioning algorithm module...may merge suitable type representatives of the ordered dependence relation into a single type representative to help reduce execution time and/or memory space costs in performing the dataflow analysis (col. 10, lines 47-60)" and "nodes 510, 520, as nodes of a strongly-connected component, are therefore collapsed into a single node... and directed edges...are removed to form a collapsed

> dependence graph," col. 12, lines 36-52). The claims merely recite; "pruning local graphs representing local problems" without specifically reciting how such pruning is performed. Ruf's merging into a single type representative by collapsing a single node dependence graph can be considered as including a reduction operation. Therefore, merging includes reducing in Ruf's portioning algorithm by merging "suitable type representatives...into a single type representative", to help "reduce execution time and/or memory space costs in performing the dataflow analysis (col. 10 lines 47-60)." In response to the applicant's statement; reducing "execution time or peak storage requirements is not the same as pruning the local graphs (page 12)," the portion is recited to indicate the result of merging." (Advisory Action, page 2)

Applicant respectfully disagrees for the following reasons.

First, the words of the claim must be given their plain meaning unless applicant has provided a clear definition in the specification. *In re Zletz*, 893 F.2d 319, 321, 13 USPQ2d 1320, 1322 (Fed. Cir. 1989); *Chef America, Inc. v. Lamb-Weston, Inc.*, 358 F.3d 1371, 1372, 69 USPQ2d 1857 (Fed. Cir. 2004) (Ordinary, simple English words whose meaning is clear and unquestionable, absent any indication that their use in a particular context changes their meaning, are construed to mean exactly what they say.) The ordinary and customary meaning of a term may be evidenced by a variety of sources, *Brookhill-Wilk 1, LLC v. Intuitive Surgical, Inc.*, 334 F.3d 1294, 1298, 67 USPQ2d 1132, 1136 (Fed. Cir. 2003), including: the claims themselves, *Process Control Corp. v. HydReclaim Corp.*, 190 F.3d 1350, 1357, 52 USPQ2d 1029, 1033 (Fed. Cir. 1999); dictionaries and treatises, *Tex. Digital Sys., Inc. v. Telegenix, Inc.*, 308 F.3d 1193, 1202, 64 USPQ2d 1812, 1818 (Fed. Cir. 2002); and the written description, the drawings, and the prosecution history, see, e.g., *DeMarini Sports, Inc. v. Worth, Inc.*, 239 F.3d 1314, 1324, 57 USPQ2d 1889, 1894 (Fed. Cir. 2001). If extrinsic reference sources, such as dictionaries, evidence more than one definition for the term, the intrinsic record must be consulted to identify which of the different possible definitions is most consistent with applicant's use of the terms. *Brookhill-Wilk 1*, 334 F. 3d at 1300, 67 USPQ2d at 1137; see also *Renishaw PLC v. Marposs Societa" per Azioni*, 158 F.3d 1243, 1250, 48 USPQ2d 1117, 1122 (Fed. Cir. 1998) ("Where there are several common meanings for a claim term, the patent disclosure serves to point away from the improper meanings and toward the proper meanings."). If more than one extrinsic definition is consistent with the use of the words in the intrinsic record, the claim terms may be construed to encompass all consistent meanings. *Tex. Digital,* 308 F.3d at 1203, 64 USPQ2d at 1819. See also *Rexnord Corp. v. Laitram Corp.*, 274 F.3d 1336, 1342, 60 USPQ2d 1851, 1854 (Fed. Cir. 2001)(explaining the court's analytical process for determining the meaning of disputed claim terms); *Toro Co. v. White Consol. Indus., Inc.*, 199 F.3d 1295, 1299, 53 USPQ2d 1065, 1067 (Fed. Cir. 1999)("[W]ords in patent claims are given their ordinary meaning in the usage of the field of the invention, unless the text of the patent makes clear that a word was used with a special meaning."). Here, the plain meaning of the word "prune" is unquestionably clear. Its meaning is totally opposite of "merge". The following definitions are obtained from the Webster's II New College Dictionary, published by Houghton Mifflin Company, 1995:

Prune –vt 2. To remove or cut out as unnecessary. 3. To reduce.

Merge –vt 1. To cause to be united or gradually absorbed in stages. 2. To combine, as sets of data.

As clear from the above definitions, "pruning" is almost the opposite of "merging". Furthermore, this extrinsic reference source is consistent with the applicant's use of the terms in the specification. Applicant's specification clearly distinguishes "pruning" and "merging". (See, for example, Specification, paragraph [0034]). Therefore, the claim terms must be construed to encompass all of these consistent meaning. *Tex.Digital id.* at 1819. The Examiner's argument that "Ruf's merging into a single type representative by collapsing a single node dependence graph can be considered as including a reduction operation" is flawed. "Merging", by nature, requires combining the elements or components together. Merging A, B, and C may result in a single element D, but that single element D is still a combination of A, B, and C. In contrast, pruning A, B, and C may result in A and B, A and C, A and C, A, B, or C which is not a combination of A, B, and C.

Second, the specification must be reviewed to determine "whether the presumption of ordinary and customary meaning is rebutted." *Tex. Digital*, 308 F.3d at 1204. "The presumption will be overcome where the patentee, acting as his own lexicographer, has set forth a definition for the term different from its ordinary and customary meaning or where the patentee has disavowed or disclaimed scope of coverage, by using words or expressions of manifest exclusion or restriction, representing a clear disavowal of claim scope." *International Rectifier Corp. v. IXYS Corp.*, 361 F.3d 1363, 1368, 70 USPQ2d 1209, 1214 (Fed. Cir. 2004). Claims should be interpreted consistently with the specification, which provides content for the proper construction of the claims because it explains the nature of the patentee's invention. See *Renishaw P.L.C. v. Marposs Societa Per Azioni*, 158 F.3d 1243 (Fed. Cir. 1998). During patent examination, the pending claims must be "given the broadest reasonable interpretation consistent with the specification". See MPEP 2111. Here, the specification clearly distinguishes "pruning" from "merging" (See, for example, paragraphs [0047], [0051]; Figures 3, 4A, and 4B). "Pruning" as used in the specification means "reduce". The specification further describes how the graphs are pruned. Pruning the graphs include applying a shrinking transform. An example of shrinking is to remove an edge (See, for example, Specification, Figures 4A and 4B). Removing is clearly different than "merging".

Third, Applicant's argument is not just about the difference between the words "pruning" and "merging". The rejected claims recite "pruning local graphs" where the local graphs represent local problems which correspond to separately compilable components. Ruf merely discloses transforming a dependence relation by merging depending types in the dependence relation into a single type. The dependence relation may be represented by a dependence graph, but a dependence graph is not the same as local graphs.

In the Advisory Action dated April 6, 2005, the Examiner further states that:

> "In response, there are two reasons that applicant's statement; a function is not separately compilable, is not persuasive. First, a unit such as a procedure (i.e. function, method), subroutine, class, or database table is the smallest separately compilable element of code. The simples example can be the main() function; void main() {cout<<"test\n;}. This main function is "separately compilable." Therefore, the applicant's argument is technically incorrect. Second, the instant specification clearly recites that: a "translation unit is a subroutine, a function, or any other separately compilable software entity (Specification, paragraph 0002)" and "Local inter-procedural problems are constructed fore each translation unit, reduced, and merged together into a global problem to be solved. Local solutions are derived from the global solution, and used to optimize each translation unit when it is recompiled (specification, paragraph 0034)." Therefore, as the specification supports that a function is separately

> compilable, the applicant's argument; a function is not separately compilable, is contradictory." (Advisory Action, page 2.)

Applicant respectfully disagrees for the following reasons.

First, the Examiner takes Applicant's arguments out of context. Applicant's statement that "a function, by itself, cannot be compiled" is in the context of Ruf's dataflow analysis. Applicant argued earlier that "[t]he functions, if present, exist in the same program, not as separately compilable components," in the context that Ruf's dataflow analysis is used to approximate the expected run-time behavior of a program.

Second, the Examiner seems to avoid addressing Applicant's argument that dataflow analysis dependencies may be determined for types representing functions, which has nothing to do with local graphs representing local problems corresponding to separately compilable components. The Examiner contends that a "type" in Ruf corresponds to a separately compilable component because it also represents a function, but the Examiner failed to establish the relationship between the "type" and the "local graphs".

In the Advisory Action dated April 6, 2005, the Examiner further states that:

> "In response, the claims do not define local problems/graphs (if the applicant means the local graph in claims is somewhat different form the conventional local graph in a dataflow analysis). As admitted in the specification, the global information related to the translations units, which are separately compilable, is collected in an inter-procedural analysis phase (specification, page 1). The applicant states that the local graphs correspond to separately compilable components above. As shown above, Ruf discloses a separately compilable unit, a function. Ruf's directed acyclic graph corresponding to a partial ordering of types, which represents functions (col. 10 lines 1-10), can be considered as a local graph. Therefore, Ruf discloses that values of the local graph form a lattice under a partial ordering." (Advisory Action, page 2.)

Applicant respectfully disagrees for the following reasons.

First, as discussed above, claims should be interpreted consistently with the specification, which provides content for the proper construction of the claims because it explains the nature of the patentee's invention. See *Renishaw P.L.C. v. Marposs Societa Per Azioni*, 158 F.3d 1243 (Fed. Cir. 1998). Applicant does not have to define local problems/ graphs in the claim because the specification provides support and interpretation of what they are. The specification provides the contextual interpretation for local graphs and local problems used in Inter-procedural analysis (IPA). IPA is a phase in a compilation process to analyze the entire program and collect global information related to the translations units (See, for example, paragraph [0002]. As supported in the specification, "the local graph pruning module 142 and the IPA solver 144 are used to optimize the IPA process for separately compilable software entities . . . Local inter-procedural problems are constructed for each translation unit, reduced, and merged together into a global problem to be solved. Local solutions are derived from the global solution, and used to optimize each translation unit when it is recompiled . . . The approach is based on graph theory and discrete mathematics with concepts in partial ordering, lattice, and lattice attributes." (See, Specification, paragraph [0034]). In contrast, Ruf deals with a totally different problem. Ruf deals with dataflow analyses which generate a model of every program quantity of interest, such as each variable, expression, or storage location (Ruf, col. 1, lines 22-26). Dataflow analysis, therefore, is not the same as inter-procedural analysis.

Second, the Examiner merely uses words and references in Ruf which have nothing to do with each other and which are not related, as a group, to the claimed invention. For example, the Examiner failed to establish, among other things: (1) the relationship between "type" and "directed acyclic graph", (2) the existence of a transfer function in each of the edges in the directed acyclic graph. Even assuming that Ruf discloses (1) "type" corresponding to a separately compilable component, and (2) "directed acyclic graph" corresponding to a partial ordering of types, this does not show that Ruf discloses all the elements as recited in independent claims 1, 15, and 29. In other words, the Examiner has not shown that, among other things, Ruf shows (1) pruning the directed acyclic graphs, (2) the direct acyclic graphs having edges each of which has a transfer function, etc.

Therefore, Applicant believes that independent claims 1, 15, 29 and their respective dependent claims are distinguishable over the cited prior art references.

VIII. CONCLUSION

Applicant respectfully requests that the Board enter a decision overturning the Examiner's rejection of all pending claims, and holding that the claims are neither anticipated nor rendered obvious by the prior art.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: May 20, 2005

[illegible]

THINH V. NGUYEN
Reg. No. 42,034

12400 Wilshire Blvd., 7th Floor
Los Angeles, CA 90025-1026
(714) 557-3800

IX. CLAIMS APPENDIX

The claims of the present application which are involved in this appeal are as follows:

1. (original) A method comprising:

pruning local graphs representing local problems, the local problems corresponding to separately compilable components in a software program, each of the local graphs having edges and vertices, each edge having a transfer function, each vertex having a value, values of each of the local graph forming a lattice under a partial ordering.

2. (original) The method of claim 1 wherein pruning the local graphs comprising:

associating a use attribute to each one of the vertices in each of the local graphs, the use attribute being asserted for each vertex reachable from a named vertex;

associating an affect attribute to each one of the vertices in each of the local graphs, the affect attribute is asserted for a vertex if a named vertex is reachable from the former vertex; and

pre-solving a subgraph of each of the local graphs, the subgraph including subgraph edges, each of the subgraph edges connecting a tail vertex to a head vertex, the tail vertex having a negated use attribute.

3. (original) The method of claim 2 wherein pruning the local graphs further comprising:

shrinking the local graphs.

4. (original) The method of claim 3 further comprising solving a global problem to optimize a recompilation of the separately compilation components by an inter-procedural analysis (IPA) solver, the global problem being represented by a global graph formed from the pruned local graphs.

5. (original) The method of claim 4 wherein pruning the local graphs further comprising:

determining final edges and vertex values of the local graphs to be sent to IPA solver; and

sending the final edges and vertex values to the IPA solver, the final edges and vertex values forming the pruned local graphs.

6. (original) The method of claim 2 wherein associating the use attribute comprises:

negating use attributes for all vertices in the local graph; and

invoking a mark use operation on u for each named vertex u in the local graph.

7. (original) The method of claim 6 wherein invoking the mark use operation on u comprises:

asserting the use attribute associated with u if the use attribute is negated; and

recursively invoking the mark use operation on v for each edge connecting the named vertex u to a vertex v.

8. (original) The method of claim 2 wherein associating the affect attribute comprises:

negating use attributes for all vertices in the local graph;

invoking a mark affect operation on y for each named vertex y in the local graph.

9. (original) The method of claim 8 wherein invoking the mark affect operation on y comprises:

asserting the use attribute associated with y if the use attribute is negated; and

recursively invoking the mark affect operation on x for each edge connecting the vertex x to a named vertex y.

10. (original) The method of claim 2 wherein pre-solving the subgraph comprises:

finding a greatest fix-point solution to the subgraph.

11. (original) The method of claim 3 wherein shrinking comprises:

removing an incoming edge having a head value of a lattice-bottom.

12. (original) The method of claim 3 wherein shrinking further comprises:

transforming a subgraph having first and second edges, the first and second edges having first and second functions, the first edge connecting a first vertex to an anonymous vertex having a value v, the second edge connecting the anonymous vertex to a second vertex having a value w.

13. (previously presented) The method of claim 12 wherein transforming comprises:

removing the anonymous vertex;

removing the first and second edges;

adding a third edge having a third function and connecting the first and second vertices, the third function being combined by the first and second functions; and

changing value of the second vertex to a lattice meet of the second function of the value v and the value w.

14. (original) The method of claim 5 wherein determining the final edges and vertex values comprises:

determining each of the final edges as edge having asserted use and affect attributes for tail and head vertices, respectively; and

eliding each of the vertex values having a top value.

15. (original) A computer program product comprising:

a machine useable medium having computer program code embedded therein, the computer program product having:

computer readable program code to prune local graphs representing local problems, the local problems corresponding to separately compilable components in a software program, each of the local graphs having edges and vertices, each edge having a transfer function, each vertex having a value, values of each of the local graph forming a lattice under a partial ordering.

16. (original) The computer program product of claim 15 wherein the computer readable program code to prune the local graphs comprising:

computer readable program code to associate a use attribute to each one of the vertices in each of the local graphs, the use attribute being asserted if there is an edge connecting a named vertex to the each one of the vertices;

computer readable program code to associate an affect attribute to each one of the vertices in each of the local graphs, the affect attribute is asserted if there is an edge connecting the each one of the vertices to a named vertex; and

computer readable program code to pre-solve a subgraph of each of the local graphs, the subgraph including subgraph edges, each of the subgraph edges connecting a tail vertex to a head vertex, the tail vertex having a negated use attribute.

17. (original) The computer program product of claim 16 wherein the computer readable program code to prune the local graphs further comprising:

computer readable program code to shrink the local graphs.

18. (original) The computer program product of claim 15 further comprising:

computer readable program code to solve a global problem to optimize a recompilation of the separately compilation components by an inter-procedural analysis (IPA) solver, the global problem being represented by a global graph formed from the pruned local graphs.

19. (original) The computer program product of claim 18 wherein the computer readable program code to prune the local graphs further comprising:

computer readable program code to determine final edges and vertex values of the local graphs to be sent to IPA solver; and

computer readable program code to send the final edges and vertex values to the IPA solver, the final edges and vertex values forming the pruned local graphs.

20. (original) The computer program product of claim 16 wherein the computer readable program code to associate the use attribute comprises:

computer readable program code to negate use attributes for all vertices in the local graph;

computer readable program code to invoke a mark use operation on u for each named vertex u in the local graph.

21. (original) The computer program product of claim 19 wherein the computer readable program code to invoke the mark use operation on u comprises:

computer readable program code to assert the use attribute associated with u if the use attribute is negated; and

computer readable program code to recursively invoke the mark use operation on v for each edge connecting the named vertex u to a vertex v.

22. (original) The computer program product of claim 16 wherein the computer readable program code to associate the affect attribute comprises:

computer readable program code to negate use attributes for all vertices in the local graph; and

computer readable program code to invoke a mark affect operation on y for each named vertex y in the local graph.

23. (original) The computer program product of claim 22 wherein the computer readable program code to invoke the mark affect operation on y comprises:

computer readable program code to assert the use attribute associated with y if the use attribute is negated; and

computer readable program code to recursively invoke the mark affect operation on x for each edge connecting the vertex x to a named vertex y.

24. (original) The computer program product of claim 16 wherein the computer readable program code to pre-solve the subgraph comprises:

computer readable program code to find a greatest fix-point solution to the subgraph.

25. (original) The computer program product of claim 17 wherein the computer readable program code to shrink comprises:

computer readable program code to remove an incoming edge having a head value of a lattice-bottom.

26. (original) The computer program product of claim 17 wherein the computer readable program code to shrink further comprises:

computer readable program code to transform a subgraph having first and second edges, the first and second edges having first and second functions, the first edge connecting a first vertex to an anonymous vertex having a value v, the second edge connecting the anonymous vertex to a second vertex having a value w.

27. (previously presented) The computer program product of claim 26 wherein the computer readable program code to transform comprises:

computer readable program code to remove the anonymous vertex;

computer readable program code to remove the first and second edges;

computer readable program code to add a third edge having a third function and connecting the first and second vertices, the third function being combined by the first and second functions; and

computer readable program code to change value of the second vertex to a lattice meet of the second function of the value v and the value w.

28. (original) The computer program product of claim 19 wherein the computer readable program code to determine the final edges and vertex values comprises:

computer readable program code to determine each of the final edges as edge having asserted use and affect attributes for tail and head vertices, respectively; and

computer readable program code to elide each of the vertex values having a top value.

29. (original) A system comprising:

a processor; and

a memory coupled to the processor to store instruction code, the instruction code, when executed by the processor, causing the processor to:

prune local graphs representing local problems, the local problems corresponding to separately compilable components in a software program, each of the local graphs having edges and vertices, each edge having a transfer function, each vertex having a value, values of each of the local graph forming a lattice under a partial ordering.

30. (original) The system of claim 29 wherein the instruction code causing the processor to prune the local graphs causes the processor to:

associate a use attribute to each one of the vertices in each of the local graphs, the use attribute being asserted if there is an edge connecting a named vertex to the each one of the vertices;

associate an affect attribute to each one of the vertices in each of the local graphs, the affect attribute is asserted if there is an edge connecting the each one of the vertices to a named vertex; and

pre-solve a subgraph of each of the local graphs, the subgraph including subgraph edges, each of the subgraph edges connecting a tail vertex to a head vertex, the tail vertex having a negated use attribute.

31. (original) The system of claim 30 wherein the instruction code causing the processor to prune the local graphs further causes the processor to:

shrink the local graphs.

32. (original) The system of claim 31 wherein the instruction code further causing the processor to:

solve a global problem to optimize a recompilation of the separately compilation components by an inter-procedural analysis (IPA) solver, the global problem being represented by a global graph formed from the pruned local graphs.

33. (original) The system of claim 32 wherein the instruction code causing the processor to prune the local graphs further causes the processor to:

determine final edges and vertex values of the local graphs to be sent to IPA solver; and

send the final edges and vertex values to the IPA solver, the final edges and vertex values forming the pruned local graphs.

34. (original) The system of claim 30 wherein the instruction code causing the processor to pre-solve the subgraph causes the processor to:

find a greatest fix-point solution to the subgraph.

35. (original) The system of claim 31 wherein the instruction code causing the processor to shrink causes the processor to:

remove an incoming edge having a head value of a lattice-bottom.

36. (original) The system of claim 35 wherein the instruction code causing the processor to shrink further causes the processor to:

transform a subgraph having first and second edges, the first and second edges having first and second functions, the first edge connecting a first vertex to an anonymous vertex having a value v, the second edge connecting the anonymous vertex to a second vertex having a value w.

37. (previously presented) The system of claim 36 wherein the instruction code causing the processor to transform causing the processor to:

remove the anonymous vertex;

remove the first and second edges;

add a third edge having a third function and connecting the first and second vertices, the third function being combined by the first and second functions; and

change value of the second vertex to a lattice meet of the second function of the value v and the value w.

38. (original) The system of claim 33 wherein the instruction code causing the processor to determine the final edges and vertex values causes the processor to:

determine each of the final edges as edge having asserted use and affect attributes for tail and head vertices, respectively; and

elide each of the vertex values having a top value.

O I P E
JC101
MAY 2 3 2005
PATENT & TRADEMARK OFFICE

TRANSMITTAL FORM *(to be used for all correspondence after initial filing)*	
Application No.	09/844,345
Filing Date	April 27, 2001
First Named Inventor	Arch D. Robison
Art Unit	2193
Examiner Name	Insun Kang
Attorney Docket Number	42390P10802
Total Number of Pages in This Submission	25

ENCLOSURES *(check all that apply)*

- [x] Fee Transmittal Form
  - [x] Fee Attached
- [ ] Amendment / Response
  - [ ] After Final
  - [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
  - [ ] PTO/SB/08
- [ ] Certified Copy of Priority Document(s)
- [ ] Response to Missing Parts/ Incomplete Application
  - [ ] Basic Filing Fee
  - [ ] Declaration/POA
  - [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s)
- [ ] After Allowance Communication to Group
- [ ] Appeal Communication to Board of Appeals and Interferences
- [x] Appeal Communication to Group (Appeal Notice, Brief, Reply Brief)
- [ ] Proprietary Information
- [ ] Status Letter
- [ ] Other Enclosure(s) *(please identify below):*

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	Thinh V. Nguyen, Reg. No. 42,034 BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
Signature	[illegible]
Date	May 20, 2005

CERTIFICATE OF MAILING/TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Typed or printed name	Tu T. Nguyen		
Signature	[illegible]	Date	May 20, 2005

Based on PTO/SB/21 (04-04) as modified by Blakely, Sokoloff, Taylor & Zafman (wlr) 06/04/2004.

OIPE JC101
MAY 2 3 2005
PATENT & TRADEMARK OFFICE

FEE TRANSMITTAL for FY 2005

Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27.

TOTAL AMOUNT OF PAYMENT	($) 500.00

Complete if Known	
Application Number	09/844,345
Filing Date	April 27, 2001
First Named Inventor	Arch D. Robison
Examiner Name	Insun Kang
Art Unit	2193
Attorney Docket No.	42390P10802

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ None ☐ Other (please identify): ______

☒ Deposit Account Deposit Account Number: 02-2666 Deposit Account Name: Blakely, Sokoloff, Taylor & Zafman LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☒ Charge fee(s) indicated below
☐ Charge fee(s) indicated below, except for the filing fee
☒ Charge any additional fee(s) or underpayment of fee(s) under 37 CFR §§ 1.16, 1.17, 1.18 and 1.20.
☒ Credit any overpayments

FEE CALCULATION

Large Entity		Small Entity			
Fee Code	Fee ($)	Fee Code	Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet.	
2053	130	2053	130	Non-English specification	
1251	120	2251	60	Extension for reply within first month	
1252	450	2252	225	Extension for reply within second month	
1253	1,020	2253	510	Extension for reply within third month	
1254	1,590	2254	795	Extension for reply within fourth month	
1255	2,160	2255	1,080	Extension for reply within fifth month	
1401	500	2401	250	Notice of Appeal	
1402	500	2402	250	Filing a brief in support of an appeal	500.00
1403	1,000	2403	500	Request for oral hearing	
1451	1,510	2451	1,510	Petition to institute a public use proceeding	
1460	130	2460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
1809	790	1809	395	Filing a submission after final rejection (37 CFR § 1.129(a))	
1810	790	2810	395	For each additional invention to be examined (37 CFR § 1.129(b))	
Other fee (specify) ______					
				SUBTOTAL (2) ($)	500.00

SUBMITTED BY Complete (if applicable)

Name (Print/Type)	Thinh V. Nguyen	*Registration No. (Attorney/Agent)*	42,034	*Telephone*	(714) 557-3800
Signature	[illegible]			*Date*	05/20/05

Based on PTO/SB/17 (12-04) as modified by Blakely, Sokoloff, Taylor & Zafman (wlr) 12/15/2004.
SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450

FEE TRANSMITTAL for FY 2005

Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27.

TOTAL AMOUNT OF PAYMENT	($) 500.00

Complete if Known	
Application Number	09/844,345
Filing Date	April 27, 2001
First Named Inventor	Arch D. Robison
Examiner Name	Insun Kang
Art Unit	2193
Attorney Docket No.	42390P10802

METHOD OF PAYMENT *(check all that apply)*

☐ Check ☐ Credit card ☐ Money Order ☐ None ☐ Other (please identify): ____________

☒ Deposit Account Deposit Account Number: 02-2666 Deposit Account Name: Blakely, Sokoloff, Taylor & Zafman LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☒ Charge fee(s) indicated below
☐ Charge fee(s) indicated below, except for the filing fee
☒ Charge any additional fee(s) or underpayment of fee(s) under 37 CFR §§ 1.16, 1.17, 1.18 and 1.20.
☒ Credit any overpayments

FEE CALCULATION

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet.	
2053	130	2053	130	Non-English specification	
1251	120	2251	60	Extension for reply within first month	
1252	450	2252	225	Extension for reply within second month	
1253	1,020	2253	510	Extension for reply within third month	
1254	1,590	2254	795	Extension for reply within fourth month	
1255	2,160	2255	1,080	Extension for reply within fifth month	
1401	500	2401	250	Notice of Appeal	
1402	500	2402	250	Filing a brief in support of an appeal	500.00
1403	1,000	2403	500	Request for oral hearing	
1451	1,510	2451	1,510	Petition to institute a public use proceeding	
1460	130	2460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
1809	790	1809	395	Filing a submission after final rejection (37 CFR § 1.129(a))	
1810	790	2810	395	For each additional invention to be examined (37 CFR § 1.129(b))	
Other fee (specify) ____________					
				SUBTOTAL (2) ($)	500.00

SUBMITTED BY Complete (if applicable)

Name *(Print/Type)*	Thinh V. Nguyen	Registration No. *(Attorney/Agent)*	42,034	Telephone	(714) 557-3800
Signature	[illegible]			Date	05/20/05

Based on PTO/SB/17 (12-04) as modified by Blakely, Sokoloff, Taylor & Zafman (wlr) 12/15/2004.
SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450